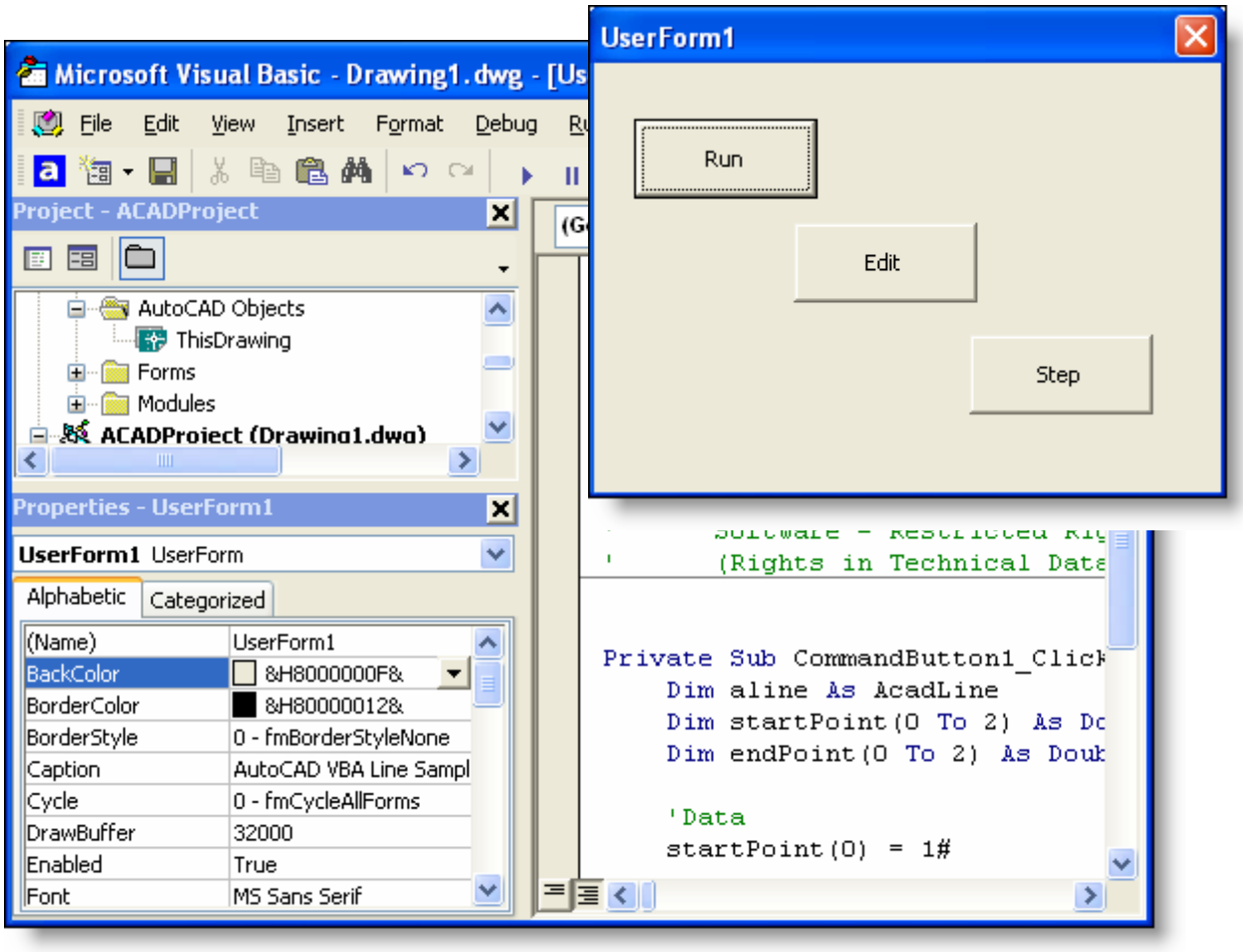
Introduction VBA for AutoCAD (Mini Guide)

This course covers these areas:

1. **The AutoCAD VBA Environment**
2. **Working with the AutoCAD VBA Environment**
3. **Automating other Applications from AutoCAD**



Contact Information

E-mail: lee_ambrosius@hyperpics.com
Web site: http://www.hyperpics.com
Web Log (Blog): http://hyperpics.blogs.com/beyond _the_ui

HyperPics
901 Estes Drive
De Pere, WI 54115

Phone/Fax: 920.338.1433

For information on how to obtain a copy of the full version of “Introduction to VBA for AutoCAD” eBook send an e-mail to the address above. The full version includes 17 Chapters with Tutorials and a total of 169 pages.

Table of Contents

Notes:

Chapter 1: History of VBA

Visual Basic for Applications (VBA) has been around for many years. It is an extension to the very popular programming language Visual Basic (VB). Although they are both the same in style of coding and working with forms and other objects they are greatly different. We will only be focusing on VBA during this course. VBA is a subset of VB that allows for custom automation of the specific application that it is built into. Many popular programs have the VBA technology built into it. Some of these applications are MS Word, MS Excel and AutoCAD start with R14.

VBA inside of AutoCAD has been a welcomed feature from the development and non-development community. VBA is a programming language that allows closer integration to the AutoCAD environment, allowing businesses to develop applications that better suit what they do and not have to worry about creating a large hole in a budget.

During this course we will be taking a look at how to navigate the Integrated Development Environment (IDE) of VBA; examine the different sections of the IDE and how to get quickly access help when you may need it for AutoCAD Objects, Properties, Methods and Events; along with learning to apply error handling and other techniques that will help you ensure that the applications that you write will run smoothly

Notes:

What Releases of AutoCAD support VBA development?

VBA is not the new kid on the block like it once was in the late 1990's when it was first introduced on the AutoCAD R14 CD as a separate install. It was classified on the AutoCAD R14 CD as a "Preview Release" of VBA which was later added as part of the install under R14.01 service pack. Since then the VBA capabilities have been in each release of AutoCAD by default with R2000 through 2005, which includes R2000i, R2002 and R2004.

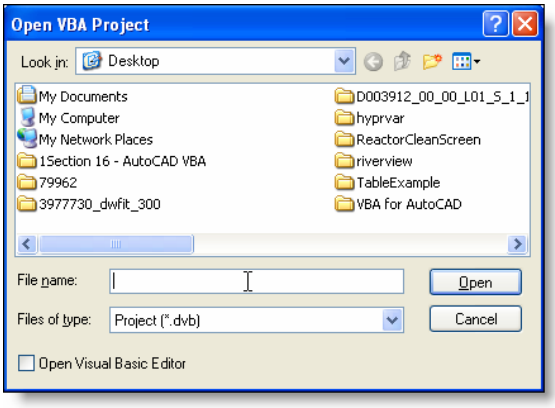
Note: Only for those that are still using AutoCAD R14

If for any reason you might still be using AutoCAD R14, you will want to make sure that you are upgraded to R14.01. To ensure that you are running AutoCAD R14.01, follow the steps below to verify what version you are currently running.

Step 1 - Launch AutoCAD and then watch for the splash screen to appear.

Step 2 - If the splash screen reads "Release 14.01" then you have the full version of the VBA components installed on the computer, but if it just reads "Release 14" then you can test to see if it is installed on your computer.

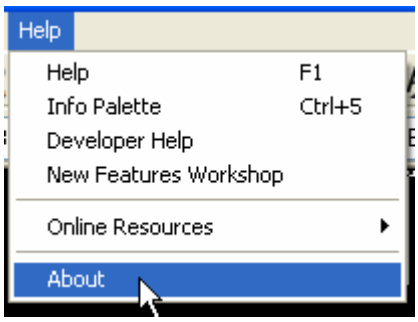
Step 3 - To test if the VBA components for AutoCAD are installed type in VBALOAD at the command line. A dialog box should appear like the one below. If the dialog box doesn't appear you should get the free upgrade disk from your local Autodesk dealer.



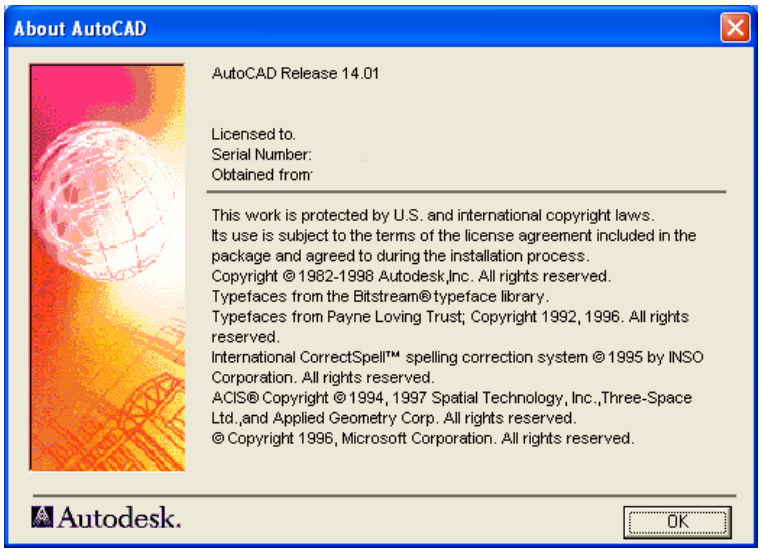
Notes:

Step 4 - If the Open VBA Project dialog box is still up on your screen, click Cancel or **ESC**ape out of it to return to the AutoCAD drawing area.

Step 5 - The other way you could check the version of AutoCAD you are currently running is to go to Help on the menubar and select About or About AutoCAD based on the version you are using.



Step 6 - Once you select the About or About AutoCAD from the Help menu bar, a dialog box should appear similar to the one below. This dialog box should display the version on it.



Notes:

Chapter 2: AutoCAD Commands for VBA

AutoCAD contains a total of six different commands that are used to perform various different tasks and functions with VBA. These four commands are: VBALOAD, VBAMAN, VBAUNLOAD and VBARUN. There are some specific Visual LISP functions that you will be using with VBA, but we will focus on those later one as they are used to enhance the interaction with Visual Lisp programs.

VBALOAD

VBALOAD is one of the more important commands for every user that chooses to work with VBA. This command loads a file into the environment, much in the same way that an AutoLISP or ObjectARX file would be loaded into AutoCAD.

> **Note:**
> **AutoCAD R14 users:** Only one project can be loaded at a single time.
>
> **AutoCAD 2000 and later users:** More than one project can be loaded at a single time.

Let's take a look at how the VBALOAD command is used. The VBALOAD command can be used in two different ways. The two ways the command can be used is in its normal state as a dialog box or from the Command Line. The two different states are controlled through the System Variable FILEDIA.

Dialog Box

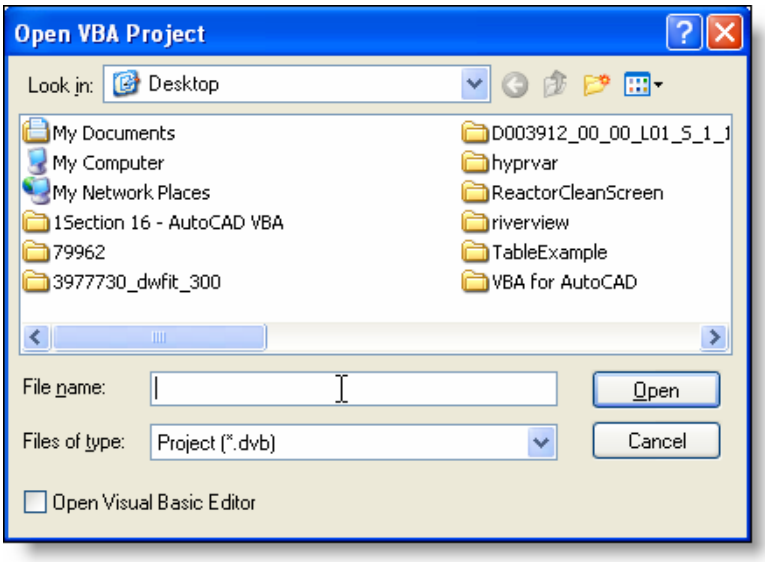
Step 1 - At the Command Line type in **VBALOAD**

- Or -

From Tools on the menu bar select **Macro>>Load Project…** If you don't have this option, don't get too worried. The Macro option shows up if you have the base AutoCAD menu loaded.

Notes:

Step 2 - With Step 1 completed correctly, you should now have a dialog box that looks like the following present on the screen.



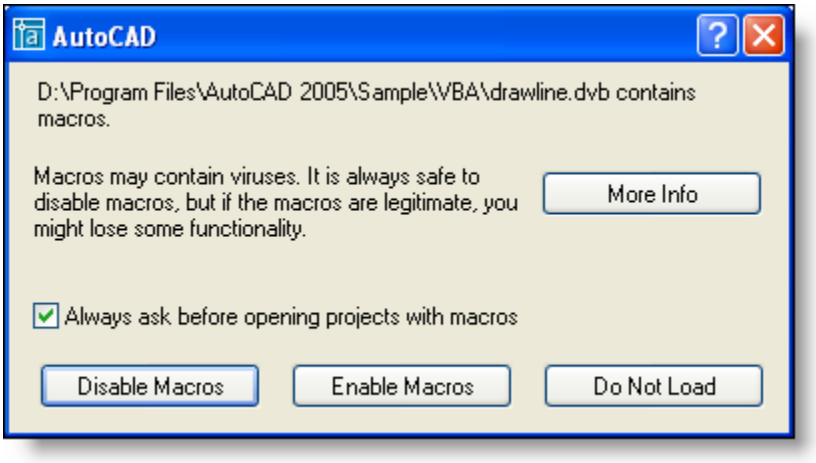
From this dialog box we can perform a couple tasks. The main task would be opening and loading a VBA project. A VBA project carries the extension of DVB and an icon that may be similar to the one below.

Proceed to the directory where AutoCAD is installed, so you can load a project into the environment. By default AutoCAD gets installed in the C:\Program Files\AutoCAD X folder, substitute X with your release of AutoCAD. If you need assistance in locating the directory, ask a co-worker or someone in your company that might normally install your applications. You might also want to try and use the Windows search feature in Windows Explorer to locate files on the machine based on a DVB extension.

Notes:

Step 3 - Once you have located the directory where AutoCAD is installed, open the folder named "Sample" and then the folder named "VBA".

Step 4 - After moving inside of the Sample\VBA folder select any one of the files that are in there, and click the Open button. The dialog box will close and do one of two different things. The first thing that might happen is that you are brought back to the drawing editor/window or you might see a message box like the one shown below.

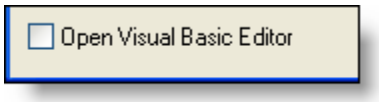


If you see the message box above you will normally click the "Enable Macros" button if you know it is a program that you are working on or have already created in the past. If you are not sure what the macro might be doing you might want to click either the "Disable Macro" or "Do Not Load" button. The "Disable Macro" button loads the project or drawing, but deactivates the macros. The "Do Not Load" button stops the loading of a VBA project or loads the drawing with the embedded macros disabled.

Step 5 - Bring back up the Open VBA Project dialog box again, (at the command line type VBALOAD).

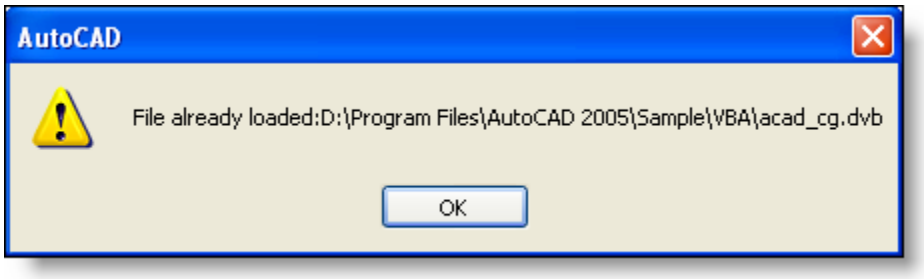
Notes:

Step 6 – If you look at the lower left corner of the dialog box there is a check box. By default this check box is unchecked. Click in the check box that is located to the left of the text "Open Visual Basic Editor" to enable the option. Now with this checked and after you select a file to open, the project will get loaded like before, but this time the Visual Basic Editor will also become active. We will talk more about the editor in just a little bit.



Step 7 – Go back to the Sample\VBA folder and select a file again, and make sure that the toggle box is checked. You will notice that a new application shows up out of nowhere. This new application is the VBA programming environment.

If a project has already been loaded in the environment, a message box like the one below will be displayed.



Notes:

VBARUN

VBARUN is the command that is used to run a macro from outside of the editor. The VBARUN command has to different syntaxes, like the VBALOAD command.

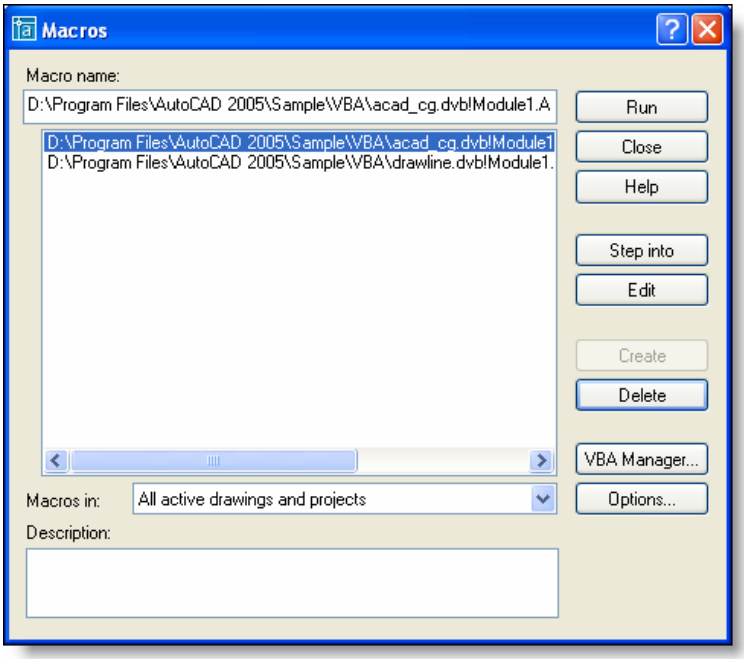
Dialog Box

Step 1 - At the Command Line type in **VBARUN**

- Or -

From Tools on the menu bar select **Macro>>Macros…**

Step 2 - If Step 1 completed correctly, a dialog box that looks similar below should be displayed on screen.



Many different tasks can be performed from this dialog box. The main task is to run a macro that is defined in one of the loaded VBA projects, but from the layout of the dialog box you can see that much more can be done here than just running a macro.

Notes:

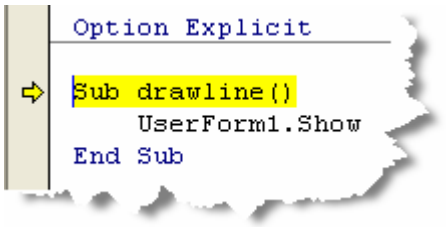
Below is a breakdown of what the buttons on the Macros dialog box perform.

Run Executes the macro that is currently highlighted in the list box which is located in the middle of the dialog box.

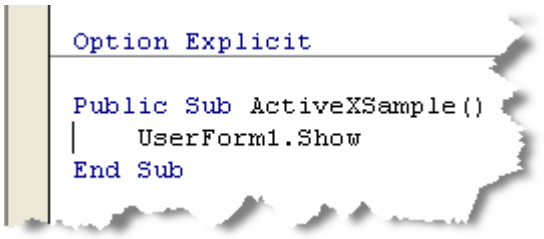
Close Exits the dialog box with out doing anything else.

Help Loads the AutoCAD help file and displays the content that is related to the Macro dialog box.

Step into Displays the Visual Basic editor and starts running the macro that is currently highlighted. The code paused at the very first line.

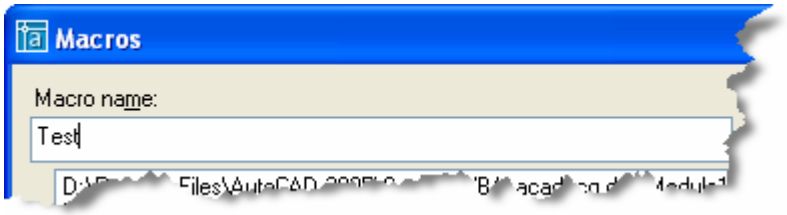


Edit Opens the Visual Basic editor to the first line of code in the macro.

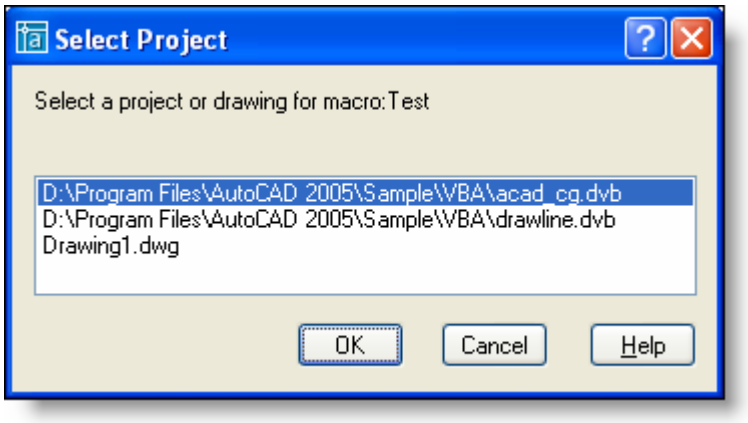


Notes:

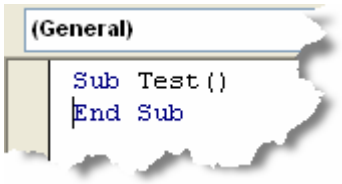
Create Creates a new macro based on the name that is entered into the Macro name field along the top of the dialog box.



After a name is provided you will then be prompted to select a loaded project to add the macro to.

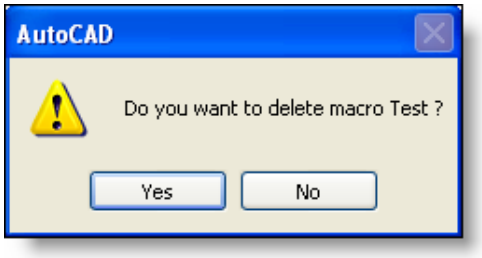


A new module called Module1 is added to the project along with the new subroutine.

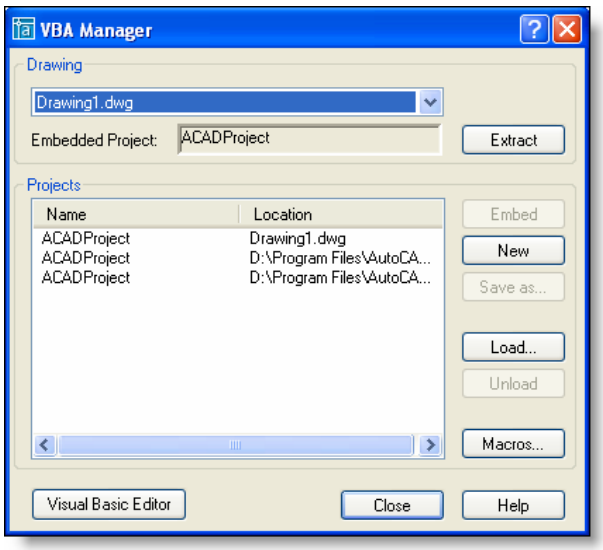


Notes:

Delete Removes the highlighted macro from the project that it is part of. A message box will be presented like the one displayed below to conform or cancel the removal of the macro.



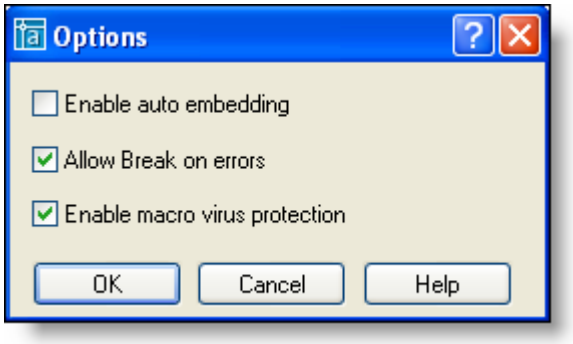
VBA Manager… Loads the VBA Manager dialog box which allows you to perform tasks that deal with things like loading and saving projects. The VBA Manager will be talked about a little bit more later on in this section. See the VBAMAN command for some further explanation and options



Notes:

Options…

The options are typically left as they come out of the box. However, there may be times when it makes sense to change these in your work environment.



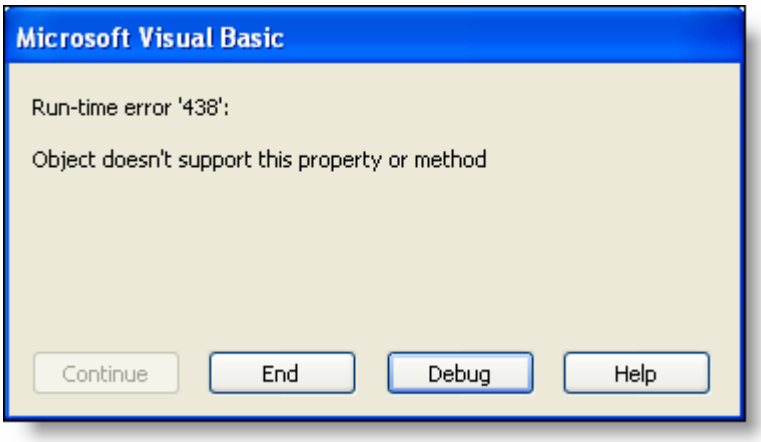
Enable auto embedding is not something you are going to want to do on all drawings, as this can lead to some issues with aging code.

Allow Break on errors might be something you want to disable to keep users out of your code when something doesn't go as planned.

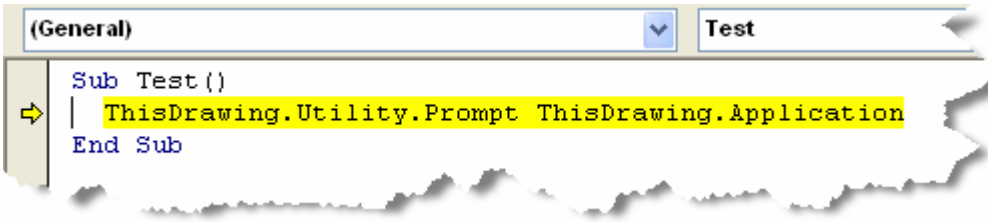
```
Sub Test()
  ThisDrawing.Utility.Prompt ThisDrawing.Application
End Sub
```

Notes:

If you ran the sample lines of code above you will get a message box like below if you leave the Allow Break on errors checked. This type of message box is not a good thing for users to see as they will usually almost always try to fix the code for you by using the Debug option.

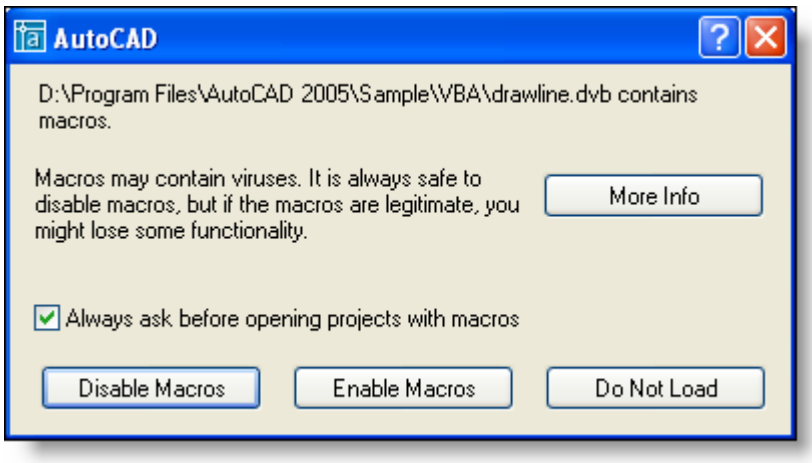


If the End button is clicked, the code execution terminates. If the Debug button is pressed, the VBA editor is loaded displaying the line of code which the error is found at.



Notes:

Enable macro virus protection when check displays the message box below, which normally appears when loading a VBA project file. If you know that you won't be getting any VBA projects in house that are not controlled you can disable this option. However, this could cause a macro that is embedded into a drawing to start running once the file has been opened and it could lead to some potential problems.



Command Line

Step 1 - Unlike the VBALOAD command, you don't need to worry about what FILEDIA System Variable is currently set to**.** In order to run **VBARUN** from the Command Line, just add a dash in front of the command name so it looks like **-VBARUN**.

Step 2 - From the Command Line you will need to enter the module name where the Sub or Function is that you desire to run with the name of the Sub or Function separated by a dot (or period). Below is an example of doing this.

Command: -vbarun
Macro name: Module1.ActiveXSample

Notes:

VBAMAN

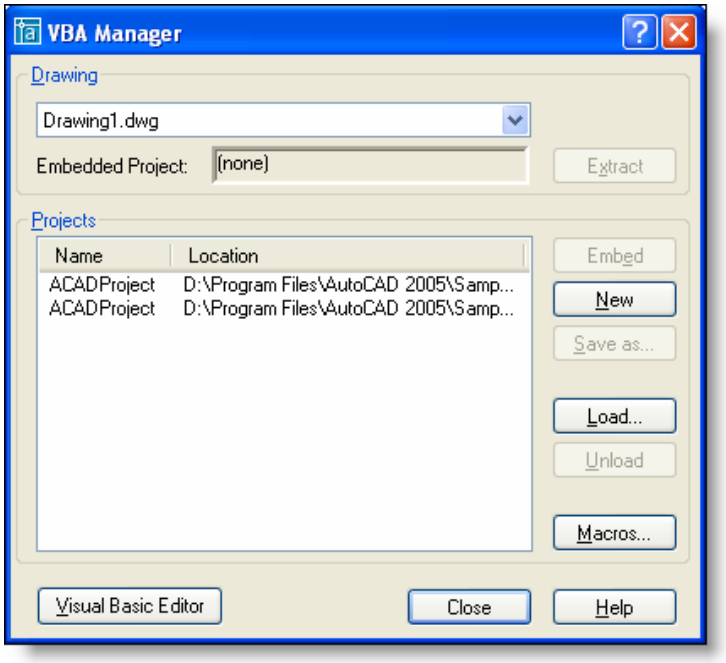
VBAMAN is the command that combines the previous commands into a single interface. This command was not originally in R14, but rather added into AutoCAD R2000 to improve workflow with VBA.

Step 1 - At the Command Line type in **VBAMAN**

- Or -

From Tools on the menu bar select **Macro>>VBA Manager…**

Step 2 - If Step 1 completed correctly, a dialog box that looks similar below should be displayed on screen.

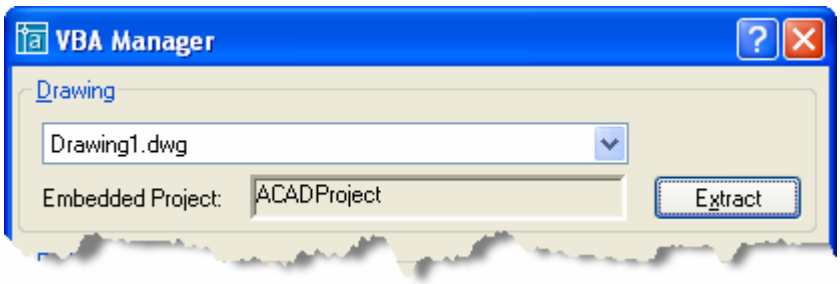


Many different tasks can be performed from this dialog box. The main task is to run a macro that is defined in one of the loaded VBA projects, but from the layout of the dialog box you can see that much more can be done here than just running a macro.

Notes:

Below is a breakdown of what the buttons on the VBA Manager dialog box perform.

Extract Removes the embedded project file from the drawing and moves it into a new project file.



Embed Makes a copy of the highlighted project and places it directly into the drawing file. This project now is part of the drawing and is accessible whenever the drawing is opened.

New Creates a new Global project which can be renamed and saved using the Save as button.

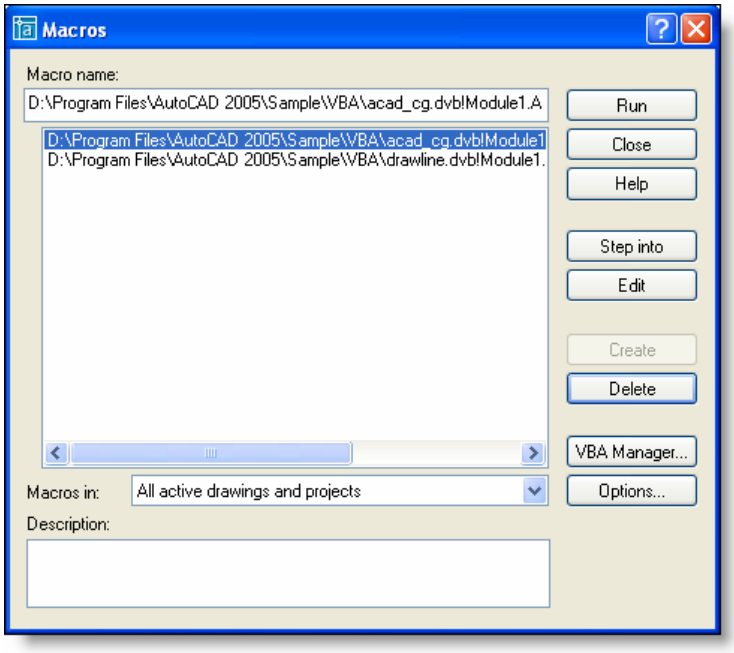
Save as… Allows you to Save out the highlighted project with a different name and location if desired.

Load… Allows the loading of a project file into the environment, much in the same way that an AutoLISP or ObjectARX file would be loaded into AutoCAD. See the VBALOAD command for some further explanation and options.

Unload Allows the unloading of a project file from the environment when it may no longer is needed. See the VBAUNLOAD command for some further explanation and options.

Notes:

Macros… Loads the Macros dialog box which allows you to perform tasks that deal with things like loading and saving projects. See the VBARUN command for some further explanation and options



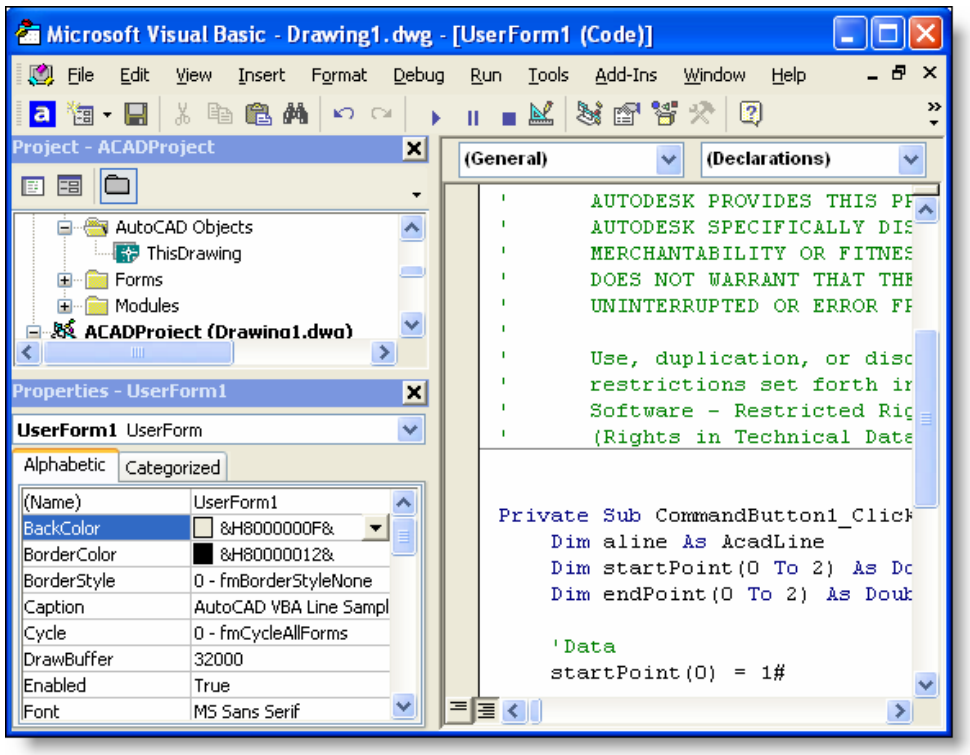
Close Exits the dialog box with out doing anything else.

Help Loads the AutoCAD help file and displays the content that is related to the VBA Manager dialog box.

Notes:

Visual Basic Editor

The Visual Basic Editor is where you will spend most of your time. It is the location where forms are added and laid out, along with non form based code modules are added. The Visual Basic Editor will be explained later on in further detail.



Notes:

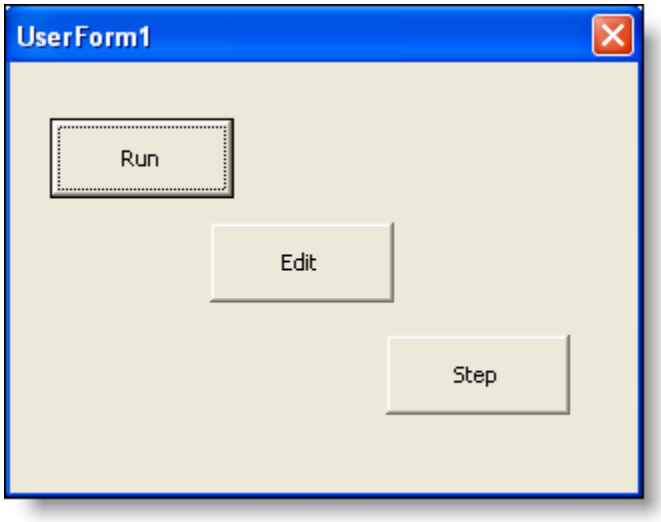
Tutorial 1: Working with Some of the AutoCAD VBA commands

Project file on CD: Chapter 2/VBARUN.dvb

In this tutorial you will use the VBALOAD command to load a VBA project into AutoCAD and then use the VBARUN command. Once done running and working with the project, use the VBAUNLOAD command to unload the project from the environment.

Step 1 - Load the project VBARUN.dvb from the CD provided with this book. Try using the different ways of loading a project with VBALOAD and VBAMAN. Don't forget that there are two different ways to use the VBALOAD command

Step 2 - Use the VBARUN command and locate the macro called ThisDrawing.Main in the listing. You can resize the dialog box if needed or select VBARUN.dvb from the Macros in drop down. Once it is highlighted in the list box the click the Run button. You should get a dialog box like the one pictured below.



Click some of the buttons and examine what is happening to the dialog box and to the AutoCAD session as you select each button. Once you are done selecting buttons choose the Close button on the dialog box, the little button with the X in the upper right hand corner of the dialog box.

Notes:

Step 3 - The remainder of this tutorial will focus on the rest of the functionality that can be found in the Macro dialog box. Run through some of the different buttons; Run, Edit, Step into and Delete.

RUN
The Run button allows for running a macro, and doesn't allow line by line debugging or evaluation of each line as it runs normally, unless breakpoints have been established.

Step A - Load up the VBARUN dialog box and select **Module1**. Now with Module1 selected choose from the drop down list box **run_me.**

Step B - Now with the two items selected press the Run button.

Step C - Notice this time the text string just appeared in the AutoCAD drawing area, and that there was no traveling to the Editor involved.

Step D - Use the Erase command to remove all text strings from the drawing area in AutoCAD.

EDIT
The Edit button allows for an easy way to jump to the code Editor and to that particular Sub or Function. This option never runs to macro unless you select the Run button from the Editors toolbar.

Step A - Load up the VBARUN dialog box and select **Module1**. Now with Module1 selected choose from the drop down list box **edit_me.**

Step B - Now with the two items selected press the Edit button. This will bring you into the Editor.

Step C - At this point you can either run the macro by pressing the blue triangle on the Editors main toolbar or you can simply exit the Editor application like you normally would exit any Windows based application.

Notes:

STEP INTO
The Step into button allows for running a macro line by line for debugging or evaluation of each line as it runs. To step completely through all the lines that are in that module press the F8 key to move to the next line; otherwise, you can press the blue square on the toolbar in the Editor to stop going through the code line by line.

Step A - Load up the VBARUN dialog box and select **Module1**. Now with Module1 selected choose from the drop down list box **step_into.**

Step B - Now with the two items selected press the Step button. This will bring you into the Editor.

Step C - Press the F8 key until you have stepped through the Subroutine, which is about 7 lines of code. After you press F8 on the End Sub line, the code will be executed. Look at AutoCAD, either a new text string was generated over an existing text object or a new text object has been created on top of an existing one.

Step D - Use the Erase command to remove all text strings from the drawing area in AutoCAD.

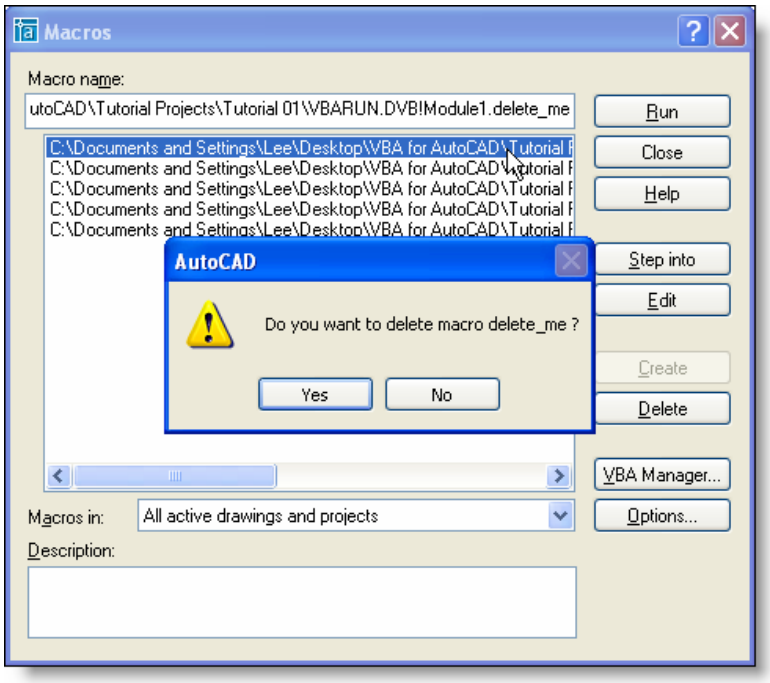
Notes:

DELETE
The Delete button allows for deleting or removing a macro.

Step A - Load up the VBARUN dialog box and select **Module1**. Now with Module1 selected chose from the drop down list box **delete_me.**

Step B - Now with the two items select press the Delete button.

Step C - You will be prompted to remove the macro. Choose OK and the **delete_me** code is then removed from your project.



Warning The Delete option can't be undone. It would be a better idea to just comment out that Sub or Function instead of completely removing it. Both would remove the reference to it under the VBARUN dialog.

Notes:

Step 4 - Try running one of the macros in the VBA project with the **–VBARUN** command, the command line version of **VBARUN**.

Step 5 – Use the VBAUNLOAD command or VBAMAN to unload the VBA project. You may want to try both methods.

Summary

This section focused on many of the commands that you will encounter and use when working with the Visual Basic customization in AutoCAD. The next section will focus on working with the Visual Basic Editor and how to access it via the VBAIDE command.

Notes:

Chapter 3: Visual Basic Editor

Loading the Visual Basic Editor

The Visual Basic for Applications Integrated Development Environment is loaded through the VBAIDE command or from the VBA Manager. The Visual Basic Editor is often referred to as IDE. The IDE is where you will spend much of your time coding, testing and debugging projects. A VBA project is a group of modules that are organized into a certain file structure. These modules include user forms, standard modules and class modules. The VBA projects that you create in this environment will only work within AutoCAD.

Step 1 - At the command line, type in **VBAIDE** to launch the Visual Basic editor. If a project is currently loaded you should see some of the components in the editor area, but if no project is loaded, you should see an empty global project loaded.

Environment Components

Every time you open up the Visual Basic editor s default project should be presented. This project contains no user forms or modules (Basic or Class), but it should include one special class module called ThisDrawing which represents the current drawing from the VBA project.

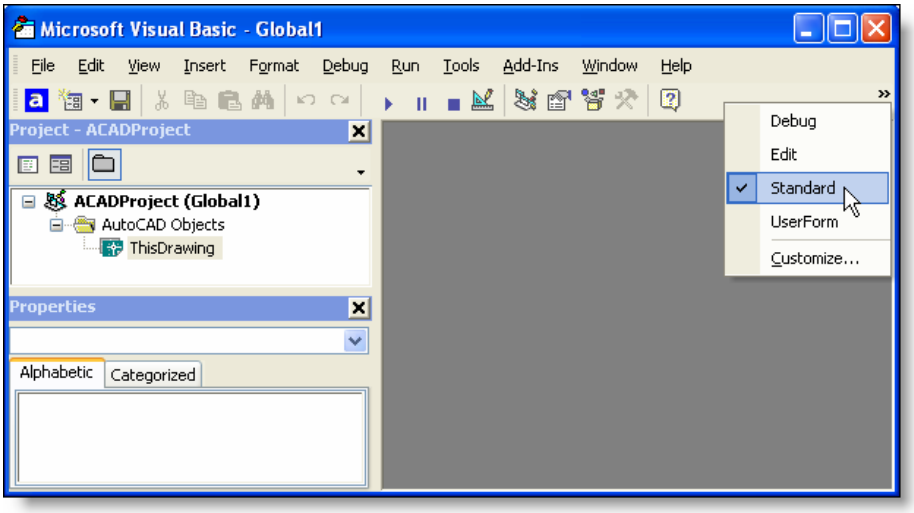
When the Visual Basic editor has completely opened up there may be several different components that could be exposed. One of these for sure will be the main application which contains menu bars, toolbars and a document area allowing code and form windows to be edited. Along with the main application you might notice many different dockable tools. There are two that are displayed by default and they are the Project Explorer and Property Page.

There may be other dockable tools that could be displayed on the screen which range from the Object Browser, a tool to view loaded reference libraries, to the Toolbox and Intermediate window. Along with all these different tools contained within the environment you will also find that there is an online help system which can be at times very helpful in the beginning when getting started.

Notes:

Main Application Window

The main application window is where all the toolbars, menu bars and the form designer/code windows are opened into.

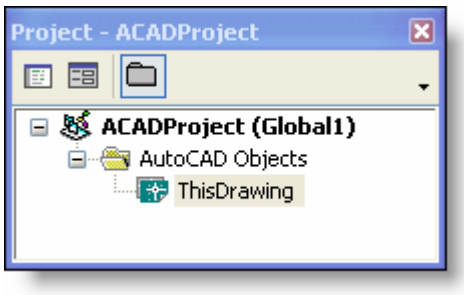


Context	Description
File	Contains controls for import/exporting modules and saving your project.
Edit	Standard Cut, Copy, other Windows document tools.
View	Contains commands for setting up the VBA environment.
Insert	Contains methods for you to add modules to your project.
Format	Contains tools for organizing controls on a form.
Debug	Tools for finding and fixing errors in the project's source code.
Run	Tools for running/testing a project.
Tools	Contains information about setting up the project and environment
Add-Ins	Contains any Add-Ins that have been loaded through the Add-In Manager.
Window	Tools for working with documents open in the environment.
Help	Methods for finding topics of a subject in a structured system.

Notes:

Project Explorer

The Project Explorer is used for displaying the files in the current project and is also used to open and view selected files. The three icons located on the top of the Project Explorer are used to view the modules in a particular type of window.



View Code Page – Used to display the code page for a user form, standard module or class module.

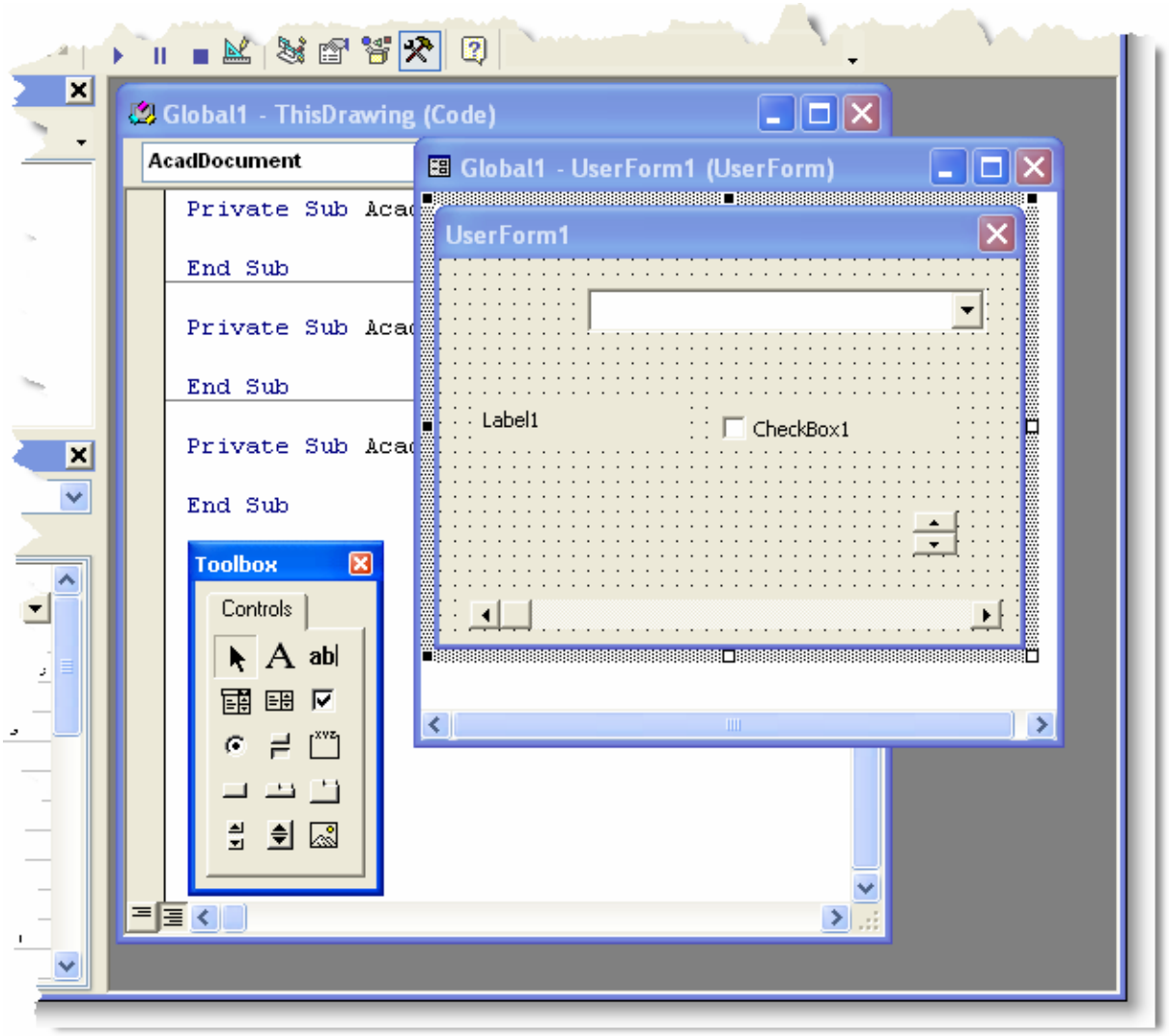
View Object – Used to display the form object which may contain controls on it.

Toggle Folders – Changes the organizational display of the Project Explorer. When the button is pressed in the battle mode is turned on.

Notes:

Project Window

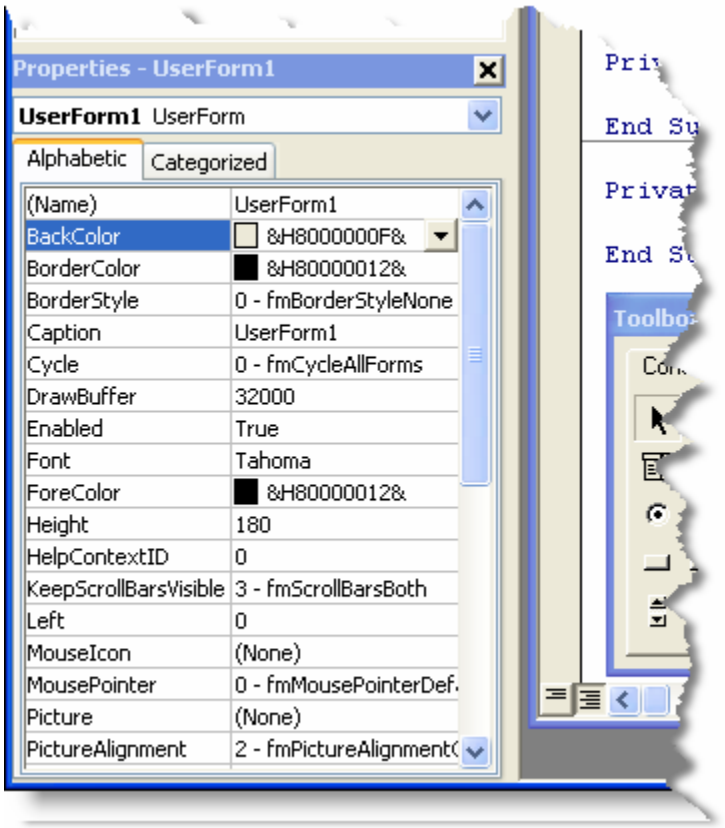
The Project Window is the area where editing/designing forms and code is input into modules. This area is usually the largest within the editor, due to the fact its where most of the work takes place.



Notes:

Properties Page/Window

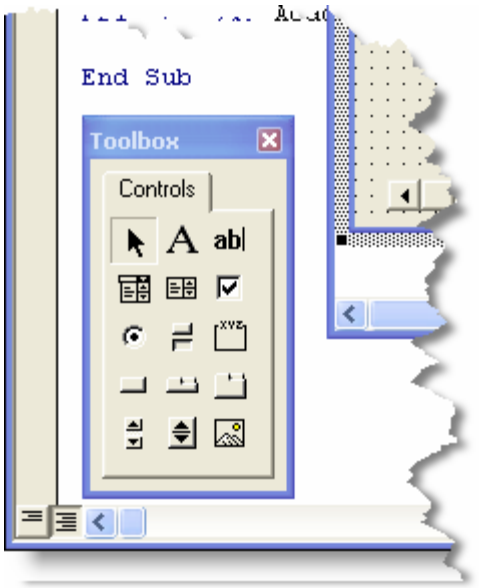
The Properties Window is the area that exposes the properties of your forms and the controls that have been placed on them. The properties can vary from control to control, but some of the common properties that are in each control is Name, Width, Height, Label/Text/Caption and appearance properties like BackColor and ForeColor. These controls can be enabled on the form and also could be invisible when the user uses the form(s). Along with the standard controls you can also add additional controls to them by setting references to their appropriate external file. Later in the course you will learn how to do this.



Notes:

Toolbox

The Toolbox is the palette with the user form tools that are currently loaded. The controls that are in the Toolbox are the default tools that will get loaded with the VBA environment every time, unless you have opened a project that has additional references built into it. To use the controls from the Toolbox all you need to do is select the tool icon and then go over to the form and drag a window to create the form.



Notes:

Tutorial 2: Working with the VBA Editor

Project file on CD: Chapter 3/UserForm.dvb

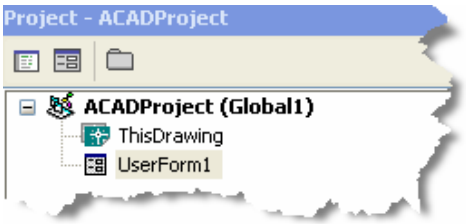
In this tutorial you learn to how to add and work with a form object in a project. Along with working with the form object properties you will also work with placing and editing controls on the form object.

Step 1 - Make sure there are no projects currently loaded into the environment by using the VBAUNLOAD or VBA Manager, unless of course you are working with an existing project.

Step 2 - Use the VBA Manager to create a new empty project at this time and display the Visual Basic editor..

Step 3 - Now that the Visual Basic editor is open, select the empty project name in the Project Window. Typically an empty project is titled ACADProject.

Step 4 - After the project title has been selected, proceed to the Insert menu bar and select UserForm. This will cause a blank form to be added to your project, and the Toolbox should also be available. You Project Window should now look similar to the image below.



Notes:

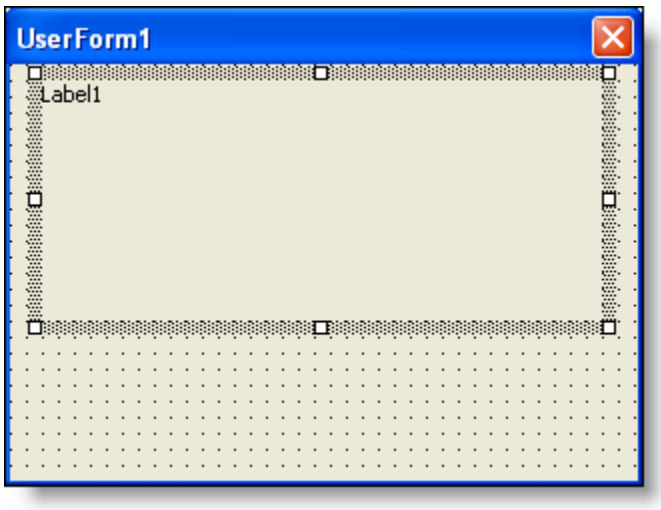
Where is My Toolbox?

If the Toolbox is not visible on screen after the User Form is added, it can be displayed by going to the View menu bar and selecting Toolbox. The Toolbox should now be visible on screen.

Note If you select on the Property, Project Explorer, or a module Code window the Toolbox will disappear for the moment, but will return when you select the Project window that contains the User Form

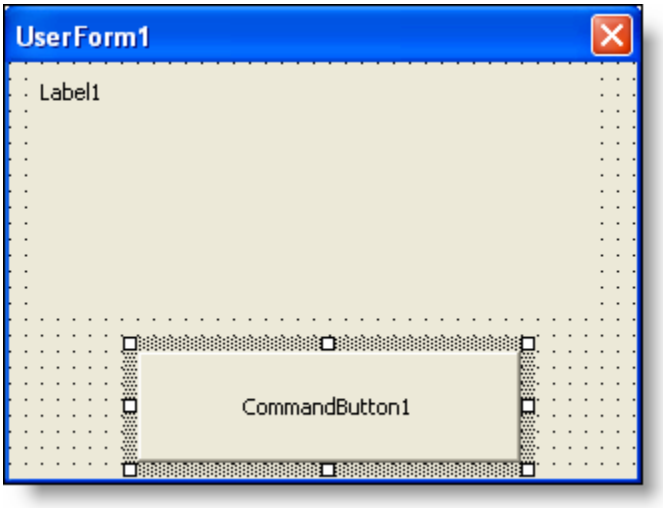
Step 5 - Now that you have added a form to the project, let's add a couple of controls to the form. Adding controls to forms can be a challenge at first, especially when trying to get everything to line up and look ideal.

Step 6 - Let's add a Label control from the Toolbox to the form. The Label control is depicted by the large black **A** and is used typically for the creation of text that is to display a message or directions on a form. By default the Label displays its control name, like Label1, Label2,... and so on.



Notes:

Step 7 - Below the Label control and near the bottom of the form add a CommandButton control. The CommandButton control is depicted by a single rectangle with a shadow in the bottom and right side of the rectangle in the Toolbox. The CommandButton control can be created with either a textual caption or display an image. Just about every Windows application uses a combination of Labels and CommandButtons. They can be found in a range of places, from the OK or Cancel buttons in a dialog box to buttons on a Toolbar. By default the CommandButton displays its control name as the caption, like CommandButton1, CommandButton2,... and so on.



Step 8 - Now that there are some controls on the User Form let's take a look at the Properties Page and Project Explorer.

Notes:

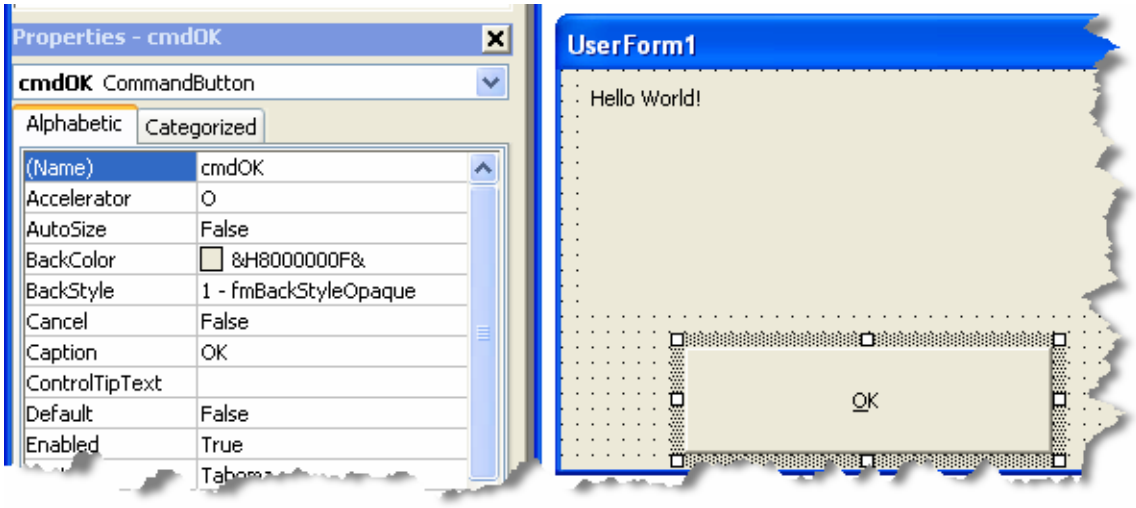
Step 9 - Left click on the Label control in the User Form to bring up its properties in the Property Page, this works very much like the Properties Palette does in the AutoCAD Drawing editor. To edit the values of the properties for the control, simply click in the box to the right of the property name. Change the Label's properties to match the following:

Name: lblHello
Caption: Hello World!

Step 10 - Follow the same steps that were used in Step 10 to edit the properties for the CommandButton. Change the CommandButton's properties to match the following:

Name: cmdOK
Caption: OK
Accelerator: O

Note The Accelerator property only exists in VBA and not in VB. In VB you would use an Ampersand (&) in front of the character you wish to use as the Accelerator Key.

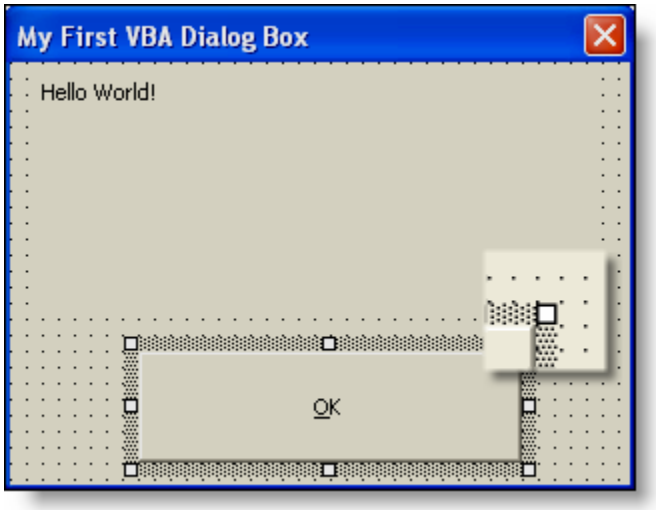


Notes:

Step 11 - Your User Form should look similar to the above image, don't worry about exact size and placement of controls. In this step you will adjust the some of the properties for the User Form. Change the UserForm's properties to match the following:

Name: frmMain
Caption: My First VBA Dialog Box

Step 12 - Make sure your dialog boxes are as small as they can be versus taking away to much from the background application and overwhelming the user of your application. So left click on the controls and the form to bring up the resizing grips. Resize the controls and form to save room on the screen and appeal.



Optimized Dialog Box:

Notes:

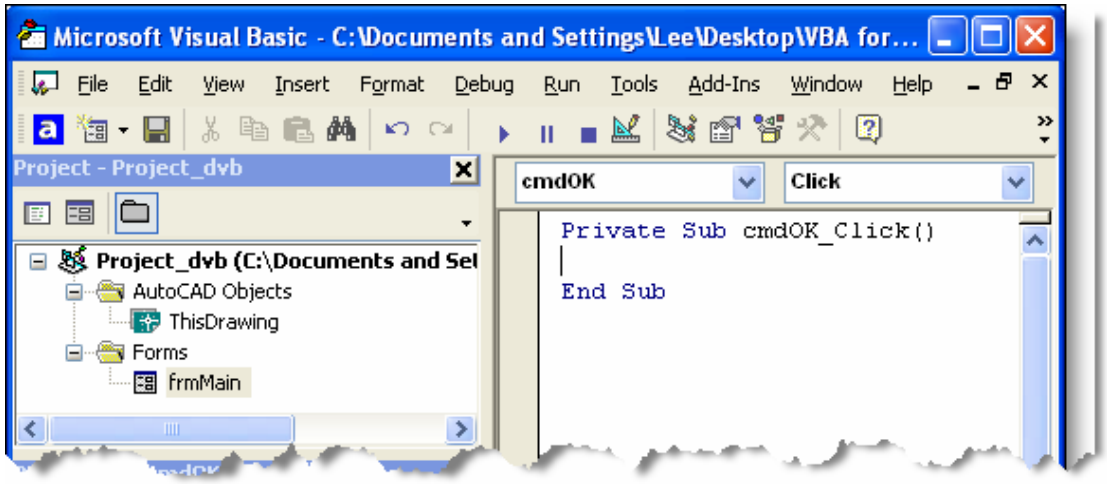
Step 13 - To run the UserForm, click on the UserForm to make sure it has focus and then select one of the following options:

Press F5 -or-
Select Run Sub/UserForm from the Run menu bar -or-
Click the Blue triangle on the Standard Toolbar

Step 14 - The only way to close out of the dialog box is by clicking on the Close button located in the upper right corner of the dialog box. In the next couple of steps you will add some code to make the OK button perform an action.

Step 15 - Make sure you stop the dialog box from running by selecting the Close button in the upper right corner or by going back to the VBAIDE and clicking the blue square Stop button.

Step 16 - Double-click on the CommandButton or right click over the CommandButton and select View Code. This will bring up the Code Window with the cursor in the Click event of the CommandButton. The screen should look similar to below.



Notes:

Step 17 - Between the two lines type-in **End**. This will cause the dialog box to simply close and unload itself. The Code Window should look similar to the code below.

```
Private Sub cmdOK_Click()
' ' Close out application
    End
End Sub
```

Step 18 - Close the Code Window and re-run the VBA project. Don't forget to make sure the UserForm has focus by clicking on it. Try to use the OK button this time again. The OK button should close the dialog box and terminate the application.

Summary

In this tutorial you should have learned how to create a new project and add a UserForm to the project. Along with adding a form, you learned how to add controls to the UserForm, adjusted properties for these controls and the form, and also added some code to get the command button to function. In later sections of this course we will add more Events and code to the project to get greater functionality from it.

Notes:

Chapter 4: Components of a VBA Project

A VBA project can contain as little as one Class module called ThisDrawing and several lines of code, to a whole set of UserForms, Modules, and Class Modules to make up a complex application. The types of Modules and other items that are in your project are based on what you are trying to build as an end solution. This section is going to detail what each component is used for and what each holds from a code standpoint.

Code Modules

Standard Modules

Contain procedures, types, and data declarations & definitions that other parts of your project reference to. You should use modules to contain code that you would like to re-use for future programs or call from multiple places in a project. This helps to avoid redundant code and re-writing code for future applications.

Class Modules

Contain code that defines a user-created object class. All AutoCAD projects begin with a class module called ThisDrawing. ThisDrawing represents the class object that defines the current AutoCAD drawing or template associated with the VBA project. Although you can create your own Class Modules with VBA, it is very rare that you would need to. It is also a lot more time consuming to create Class Modules in VBA than it is under the standard VB environment due to the lack of wizards to help make the tasks less of a headache.

Notes:

Dim

Used to Dimension (or declare) a variable as a specific data type. See the examples below.

```
Dim objLine as AcadLine
Dim strValue as String
Dim vVar
```

Note When a variable is not declared with Dim, it is automatically assigned to being a variant, but this is bad practice. It is also not recommended practice too not Dimension a variable without a data type like in the example Dim someVar.

Tip It is a good idea to adopt a good naming convention for your variables and the scope of the program that they are being used in. Variables fall into two different scopes and they are Public and Private, just like Subs and Functions. I will typically use p_ in front of a variable that I am using in a Public scope of my project. This helps to quickly identify how the value of the variable is being used.

Definition Scope - Refers to where in the program the variable can be accessed and/or modified.

Common levels:

- global: visible to the entire program
- local: visible only to the local function or with in the block the name was declared

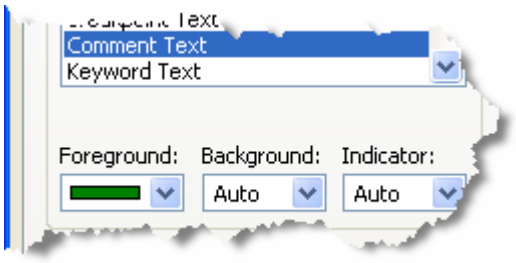
Notes:

Comments

A Comment is a string of information that is used to describe a line or series of code. The Comment provides a way to identify what is taking place at that location in the program. A Comment is denoted by an Apostrophe (') that appears in the front of the string. By default the color of a Comment in the VBA Integrated Development Environment is Green.

```
'' Close out application
```

Note There is an option under the Options dialog box within the Visual Basic editor that allows for changing the colors used in the editor for many things. If you don't like the default Green color used for comments it can be changed. To make this change go to the Tools menu bar within the Visual Basic Editor and select Options… . Make sure the Editor Format tab is active and select Comment Text from the list box and then change its color below.



=

The Equals (=) sign is for two different purposes in VBA. The Equals sign when used with a variable is for value assignment. The other purpose is for comparison to see if two values are Equal To each other, this will be covered in more depth later.

```
Dim strValue as String
strValue = "Hello World!!"

Dim nCount as Integer
nCount = 1
```

Notes:

User Form

The User Form is a dialog box interface within a project. The dialog box may contain controls that allow user interaction or simply provided useful information in the form of an alert message or instructions. Controls that can be found on a dialog box may range from buttons, sliders, images and many other common Windows controls. A project is not limited to a single form and often a dialog box might be used to call up other dialog boxes on screen.

A dialog box is usually displayed in one of two different states; Modal or Modeless. When a form is Modal which is the default behavior the user is not allowed to interact with AutoCAD while the form is displayed on the screen. Now if the form is loaded as Modeless, the user is allowed to interact with both AutoCAD and the form at the same time. Typically you want the users focus with the dialog box, but if your form is designed to collect information about what is happen in the drawing real-time you might want it on the screen to provide feedback to the user.

On the CD provided with this book is a project called FormStates.dvb and it shows how the different states for a form work. They are not very elaborate examples, but should give an understanding of the difference between Modal and Modeless though.

Step 1 - Load the project FormStates.dvb from the Chapter 4 folder on the CD provided with this book

Step 2 - Use the Macros dialog box to run the two different macros one at a time called RunAsModal and RunAsModeless. Try to interact with AutoCAD by picking on the screen and trying to run commands.

You should notice that the Modal dialog box won't allow you to do much of anything, but the Modeless dialog box should let you run commands with in the drawing editor. As commands are being run they should start appearing in the list box as a running history.

Notes:

Form Controls

Form controls are objects that have built in Properties, Methods, and Events. These objects are of a special type since they are graphically placed on a UserForm object. Below is a list of the common Form Controls that are used with VBA UserForms in AutoCAD projects.

Control Type/ Hungarian Notation	Description
Label lbl	A static control for descriptive text.
TextBox txt	A control that allows the user to enter a value in to.
ComboBox cbo or cmb	A control that can contain either preset data or a place the user can enter a value in to.
ListBox lst	A control that displays a list of values, can work with either single or multiple selection.
CheckBox chk	A control that can either represent a value of True or False, displays a Checkmark when the value is True.
OptionButton opt	A control that can either represent a value of True or False, but only one can be True within a group of OptionButtons.
ToggleButton tgl	A control that can either represent a value of True or False, button is graphically pressed in when it is set to True.
Frame fra or fam	A control that is used to represent the grouping of controls.

Notes:

Control Name	Description
CommandButton cmd or btn	A control button that is used commonly to close a dialog box or launch a new function from with in a dialog box.
TabStrip tb or tab	A control that is used to control the display of controls for a Options or Properties page, more code is required for this control than the MuiltiPage. Control is not a container object.
MultiPage mp	A control that is used to control the display of controls for a Options or Properties page, less code is required for this control than the TabStrip. Control is a container object.
Scrollbar sb	A control that allows you to add a visual status at what point you are along a specific value.
SpinButton spb	A control that allows you to scroll through a range of values, but not visually like the Scrollbar controls.
Image img	A control that displays a picture within a bounding box. Supported Image formats: *.bmp *.cur *.gif *.ico *.jpg *.wmf

Notes:

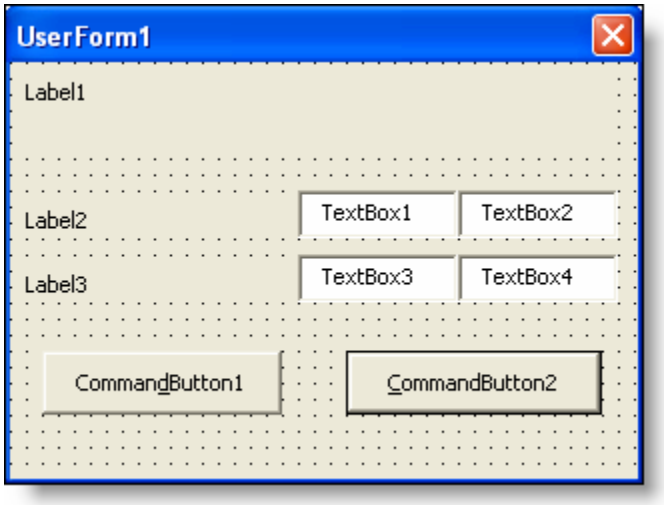
Tutorial 3: Working with Variables, Procedures and User Forms

Project file on CD: Chapter 4/DrawLine.dvb

This tutorial will focus on adding variables and custom procedures to a project. It will also use many of the things that you learned in the previous tutorials to help reinforce things as we get deeper into the Visual Basic editor and project complexity.

Step 1 - Make sure there is a blank project loaded currently in the environment by using the VBA Manager to unload projects and create a new one. Then start up the Visual Basic editor if it isn't already running.

Step 2 - Now that the Visual Basic editor is open, add a new User Form to the project. Add three Labels, two TextBoxes and two CommandButtons onto the User Form. Arrange the controls so they look are similar to the image below, but position and size of the controls are not important at this time.



Notes:

Step 3 - Change the Properties of the controls and/or form below to match the list below:

UserForm
Name: frmDrawLine
Caption: Draw Line

Label1
Name: lblDescription
Caption: This application is used to draw a line with user input typed into two Text Boxes. Input must be supplied as X and then Y format.

Label2
Name: lblStartPt
Caption: Start Point:

Label3
Name: lblEndPt
Caption: End Point:

TextBox1
Name: txtStartPtX

TextBox2
Name: txtStartPtY

TextBox3
Name: txtEndPtX

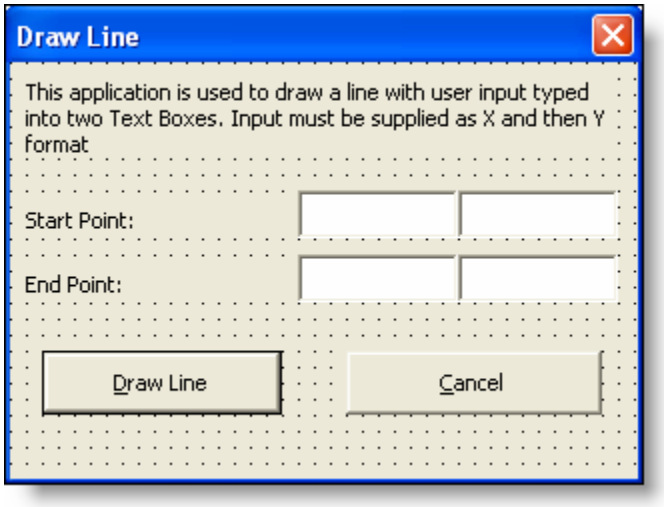
TextBox4
Name: txtEndPtY

CommandButton1
Name: cmdDrawLine
Caption: Draw Line
Accelerator: D
Default: True

CommandButton2
Name: cmdCancel
Caption: Cancel
Accelerator: C
Cancel: True

Notes:

Once you have completed the changes to the controls and the form properties, the UserForm should look very similar to the following image.

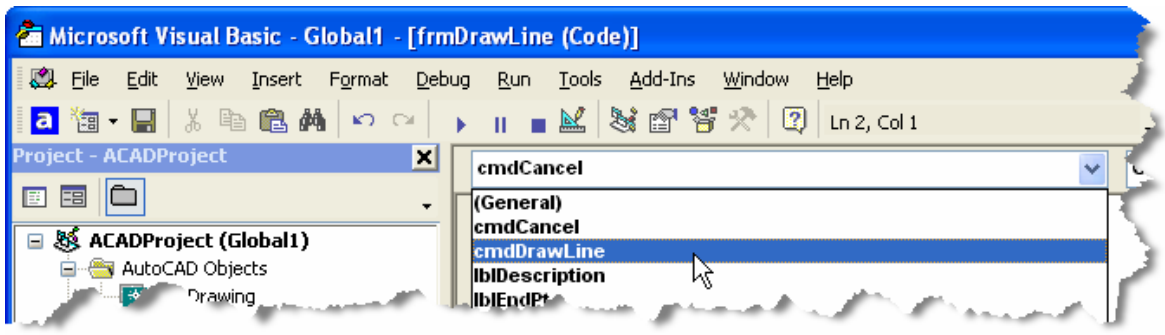


Step 4 - Double-click on the Cancel Button or right-click over the Cancel Button and select View Code. The cursor should automatically be placed between the two lines of code, type in **End** between these two lines. This will cause the dialog box to simply close and unload itself. Add a comment right above the new line of code that you just entered. Type in the comment of **'' Close out application**. The Code Window should look similar to the code below.

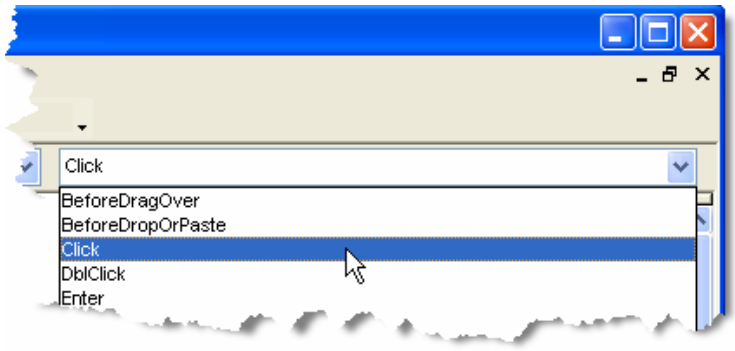
```
Private Sub cmdCancel_Click()
  ' ' Close out application
  End
End Sub
```

Notes:

Step 4 - Bring up the Code Window again for the Draw Line button this time. You can use the same steps as in the previous step, otherwise if you are already in the Code Window you can select the control or object that you want to access a Procedure from. Click on the first dropdown located near the top of the Code Window and select cmdDrawLine one of the Command button control names on the User Form.



Step 5 - Locate the dropdown list next to the one that you found the cmdDrawLine Command button in is the Procedure list for this object. These might be Procedures that were manually added or standard built-in Procedures for the object. The built-in procedures could be events that are activated based on user interaction with that control, like a click from the mouse or a keystroke from the keyboard. Open this dropdown list up and select the Click procedure. This will add the same code that you say before when you Double-clicked on the Cancel button or right-clicked over the control and selected View Code.

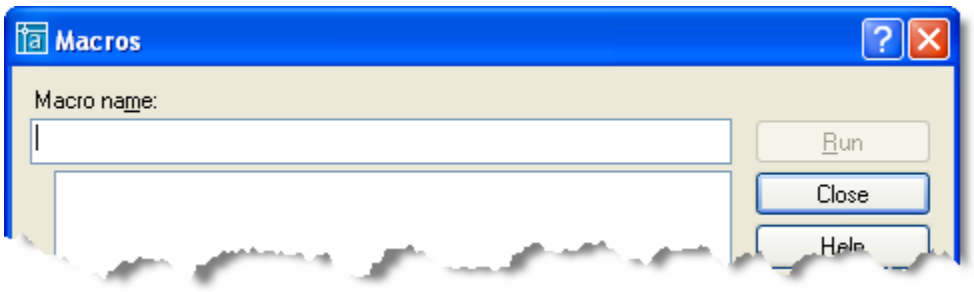


Notes:

Step 6 - Now that Click procedure has been added for the cmdDrawLine Command button add the code below. Once you are complete the Code Window should look similar to the code below.

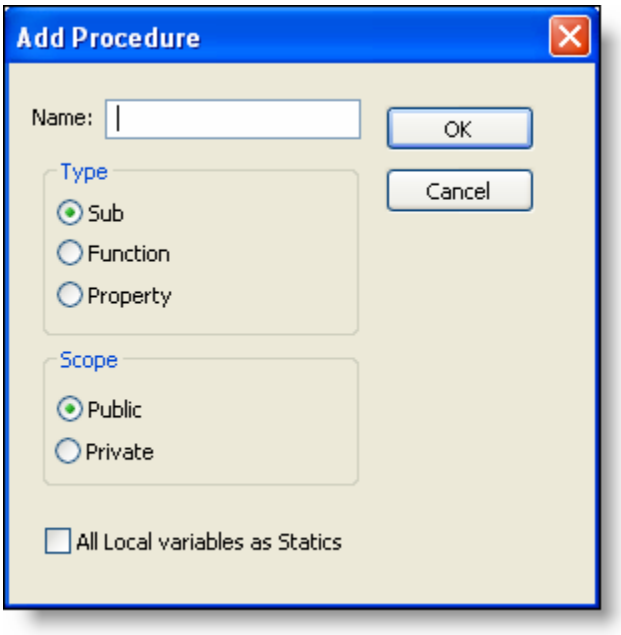
```
Private Sub cmdDrawLine_Click()
  Dim objLine As AcadLine
  Dim PT1(0 To 2) As Double, PT2(0 To 2) As Double

  ' ' Construct the two point arrays needed to draw the points for the Line object
  PT1(0) = CDbl(txtStartPtX): PT1(1) = CDbl(txtStartPtY): PT1(2) = 0#
  PT2(0) = CDbl(txtEndPtX): PT2(1) = CDbl(txtEndPtY): PT2(2) = 0#

  ' ' Draw a Line Object in Model Space for the Active Document
  Set objLine = ThisDrawing.Application.ActiveDocument.ModelSpace.AddLine(PT1, PT2)

  ' ' Perform a Zoom Extents on the Active Viewport (assuming it is Model Space)
  ThisDrawing.Application.ZoomExtents
End Sub
```

Step 7 - Switch over to AutoCAD and use the VBARUN command to try and run your application to see how it works. You should notice that there is no way to start the application from the Macro dialog box.



Notes:

Step 8 - Let's add some code to allow for the loading of the application. Close the Macros dialog box and switch back over the Visual Basic editor. Once the Visual Basic editor is visible, add a new Module to your project from the Insert menu bar within the Visual Basic editor. Double-click on the new model called Module1 within the Project Explorer. This will cause a new Code Window to come up. Go to the Insert menu bar and select the Procedure… option. This will cause the Add Procedure dialog box to appear.

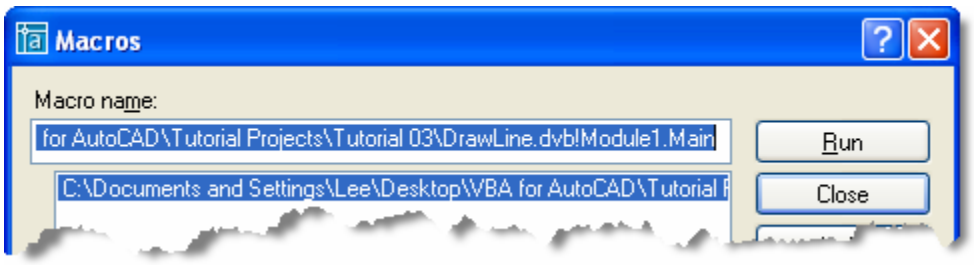


Step 9 - Enter **Main** in the name textbox. The Type should be Sub and with a Scope of Public so it can be called from anywhere. The following code should appear in the Code Window for the Module1 basic module.

```
Public Sub Main()
  ' ' Causes the Draw Line form to be displayed on the screen
  frmDrawLine.Show
End Sub
```

Notes:

Step 10 - Try to run the macro again from the Macros dialog box again. You will notice now there are macros that you can choose from this time. Select the Module1.Main procedure and then click the Run button to start the application up. The dialog box that you built in the Visual Basic editor should appear on the screen, which allows you to interact with it.



Step 11 - Type in some whole or decimal numbers for the Start and End Points. Then click the Draw Line button to see what happens. A new line should be drawn on screen and on the current layer. You can use the Properties Palette to see if the line start and end point properties match what you entered into the dialog box. Did you get a Line on the screen? If you don't see the line within the drawing, make sure to check the state of the current layer so it isn't frozen or off. You might want to create a new drawing and try again.

Summary

In this chapter you should have learned how to work with procedures, user forms and controls. The way you interacted with user forms and controls was at Design Time through the Visual Basic editor. As time goes on you will learn more about the Run Time interaction with controls and the user form. This tutorial also covered adding a procedure in a Basic code module so you can interact with the project through the VBARUN command.

Notes:

Notes: